实验一 组合逻辑设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握基础组合逻辑电路的设计和测试方法。

**二、实验内容（用Logisim或Vivado实现）**

1 基础门电路（多输入门电路、复用器等）的设计和测试；

2 基础功能模块（编码器、译码器等）的设计与测试。

**三、实验要求**

1 掌握Vivado与Logisim开发工具的使用，掌握以上电路的设计和测试方法；

2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

1、4路复用器mux41的设计和仿真

1.1设计文件及其代码分析：

###mux41.v###

module mux41(

input D0,input D1,input D2,input D3,

input [1:0] S,

output Y

);

reg temp;

always@(\*)

begin

case(S)

2'b00:temp=D0;

2'b01:temp=D1;

2'b10:temp=D2;

2'b11:temp=D3;

default:temp=D0;

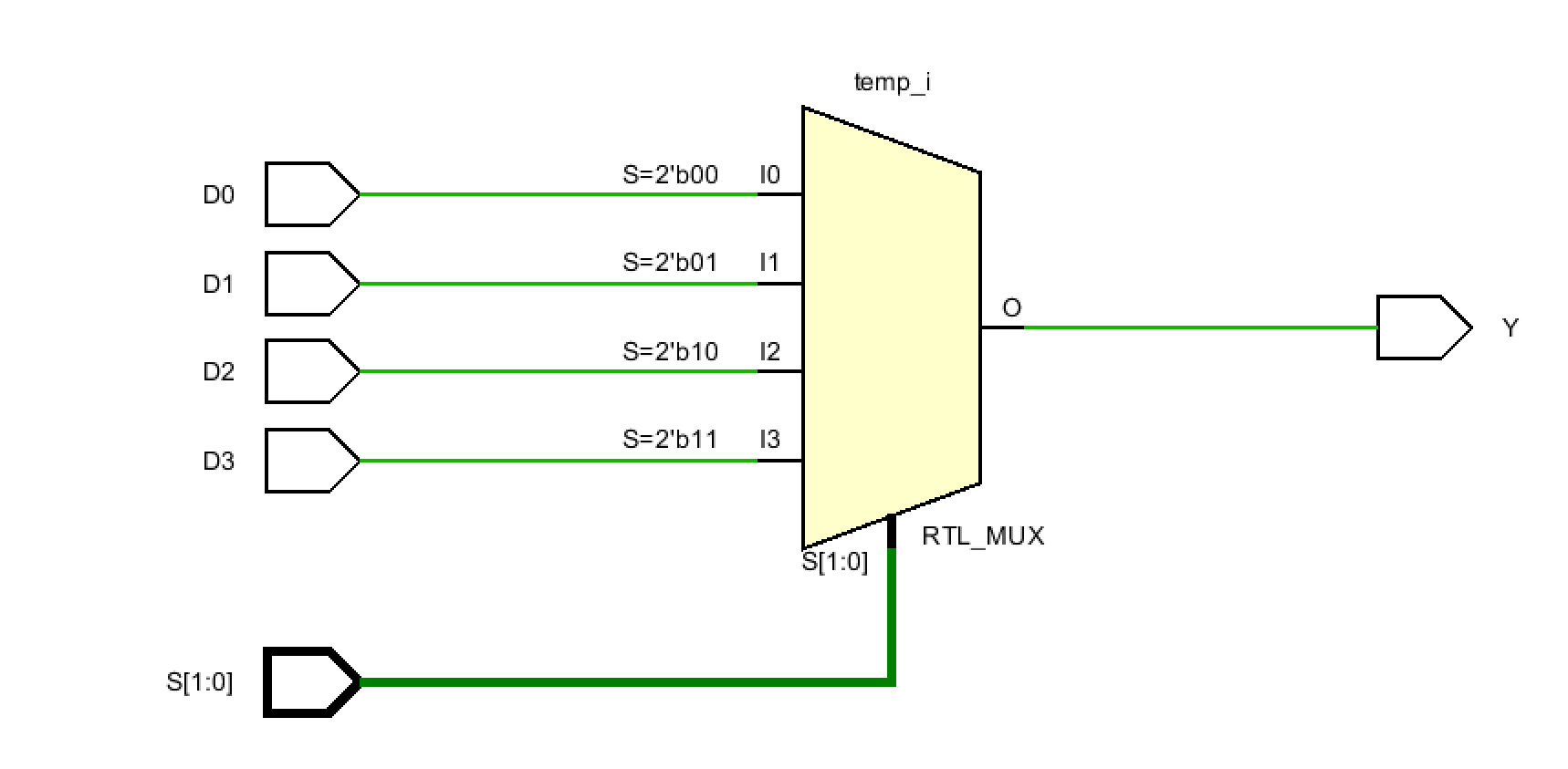
endcase

end

assign Y=temp;

endmodule

逻辑图：



代码分析：首先声明设计模块名称为mux41，声明输入变量为D0,D1,D2,D3,S，输出变量为Y，其中D0,D1,D2,D3为4个输入信号，S为宽度为2的选择信号。声明temp寄存器用来存储选择到的信号。使用always@(\*),只要任意一个输入信号发生变化，即更新temp及寄存器的值，以完成实时更新。最后使用assign语句实时把temp寄存器的值赋给Y，相当于用一个线把寄存器的值与Y连接。

1.2仿真文件：

###sim\_mux41.v###

module sim\_mux41;

reg D0,D1,D2,D3;

reg [1:0] S;

wire Y;

mux41 mux(D0,D1,D2,D3,S,Y);

initial

begin

D0=0;D1=0;D2=0;D3=0;S=2'b00;

fork

repeat(100)#10 D0 = ~D0;

repeat(50)#20 D1=~D1;

repeat(25)#40 D2=~D2;

repeat(10)#100 D3=~D3;

repeat(5)#200 S=S+1;

join

end

endmodule

逻辑图:

时序图：



代码分析：

首先声明变量并利用设计文件实例化一个mux41模块mux，然后将D0,D1,D2,D3初始化为0，选择信号初始化为b’00。然后使用fork语句同时执行以下语句：

每隔100ns把D0取反，重复10次

每隔50ns把D1取反，重复20次

每隔25ns把D2取反，重复40次

每隔10ns把D3取反，重复100次

每隔5ns把S自增1，重复200次

2、多输入门电的设计和仿真

2.1 多输入门电路的设计文件

###mul\_gate###

module mul\_gate(

input a,input b,input c,input d,input e,

output X

);

reg temp1,temp2,temp3;

always@(\*)

begin

temp1=a&(~b)&c;

temp2=d||e;

temp3=~(temp1&temp2);

end

assign X=temp3;

endmodule

逻辑仿真图：

代码分析：

首先声明设计文件名称mul\_gate，并列举出模块的输入变量a,b,c,d,e和输出变量X。

声明三个寄存器temp1，temp2，temp3，同样使用always@(\*)，使得一旦由输入信号的变化时，更新各寄存器的值。

同样使用assign语句将寄存器temp3与输出Y相连。

2.2 多输入门电路的仿真文件

###sim\_mul\_gate.v###

module sim\_mul\_gate;

reg a,b,c,d,e;

wire Y;

mul\_gate(a,b,c,d,e,Y);

initial

begin

a=0;b=0;c=0;d=0;e=0;

fork

#10 a=~a;

#25 b=~b;

#30 b=~b;

#10 c=~c;

#45 c=~c;

#65 c=~c;

#15 d=~d;

#35 d=~d;

#50 d=~d;

#60 d=~d;

#20 e=~e;

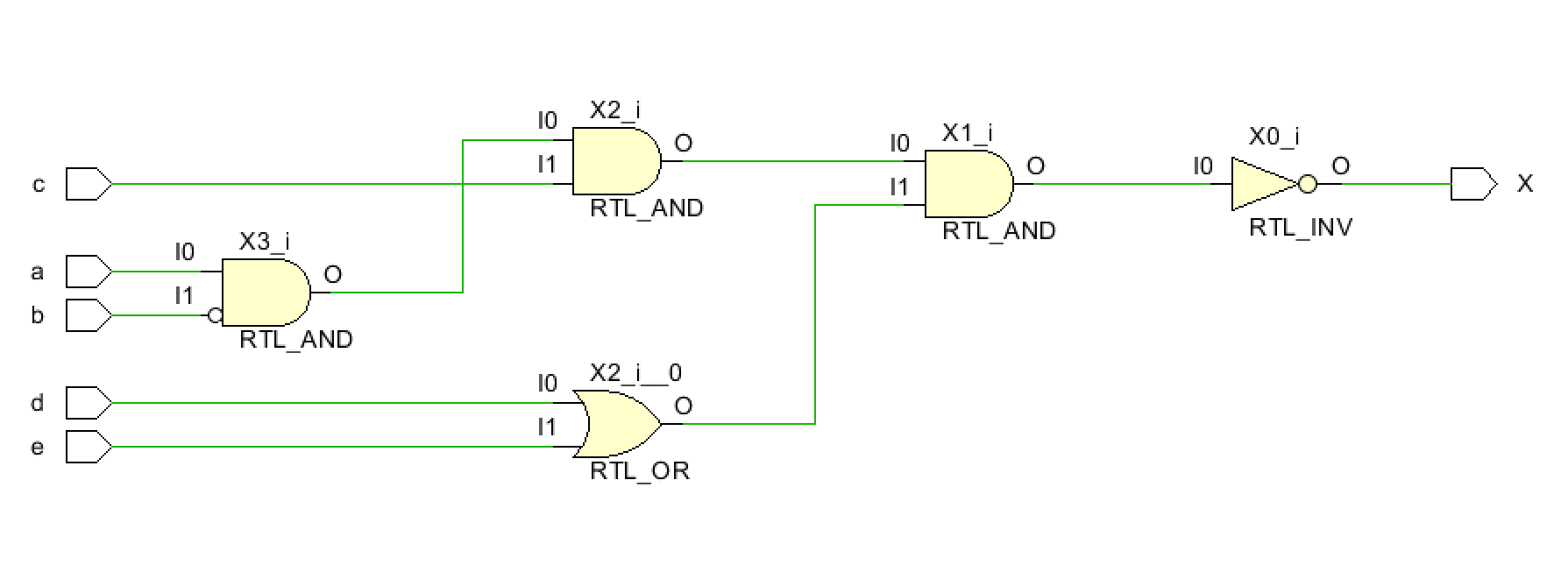
#40 e=~e;

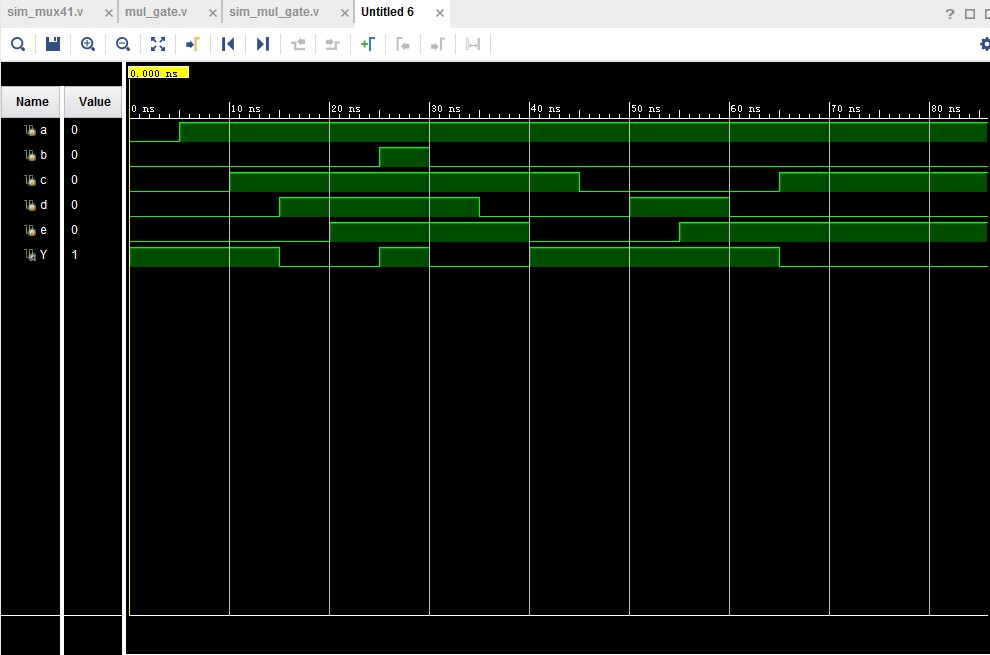
#55 e=~e;

join

end

endmodule

仿真图：

时序图

代码分析：

首先声明仿真文件模块名为sim\_mul\_gate。

定义寄存器a，b，c，d，e和输出Y。

实例化mul\_gate模块。

按照ppt中提供的时序图，将a,b,c,d,e的初值都设为零

按照ppt中提供的时序图，使用fork join语句，设计a,b,c,d,e的值的变化。

3、83编码器的设计和仿真

3.1 83编码器的设计文件

3.1.1、case结构的设计文件

module pre\_encoder83\_case(I,Y);

input I;

output Y;

wire[7:0] I;

reg[3:1] Y;

always @(I) begin

casex(I)

8'b0000\_0001:Y=3'b000;

8'b0000\_001X:Y=3'b001;

8'b0000\_01XX:Y=3'b010;

8'b0000\_1XXX:Y=3'b011;

8'b0001\_XXXX:Y=3'b100;

8'b001X\_XXXX:Y=3'b101;

8'b01XX\_XXXX:Y=3'b110;

8'b1XXX\_XXXX:Y=3'b111;

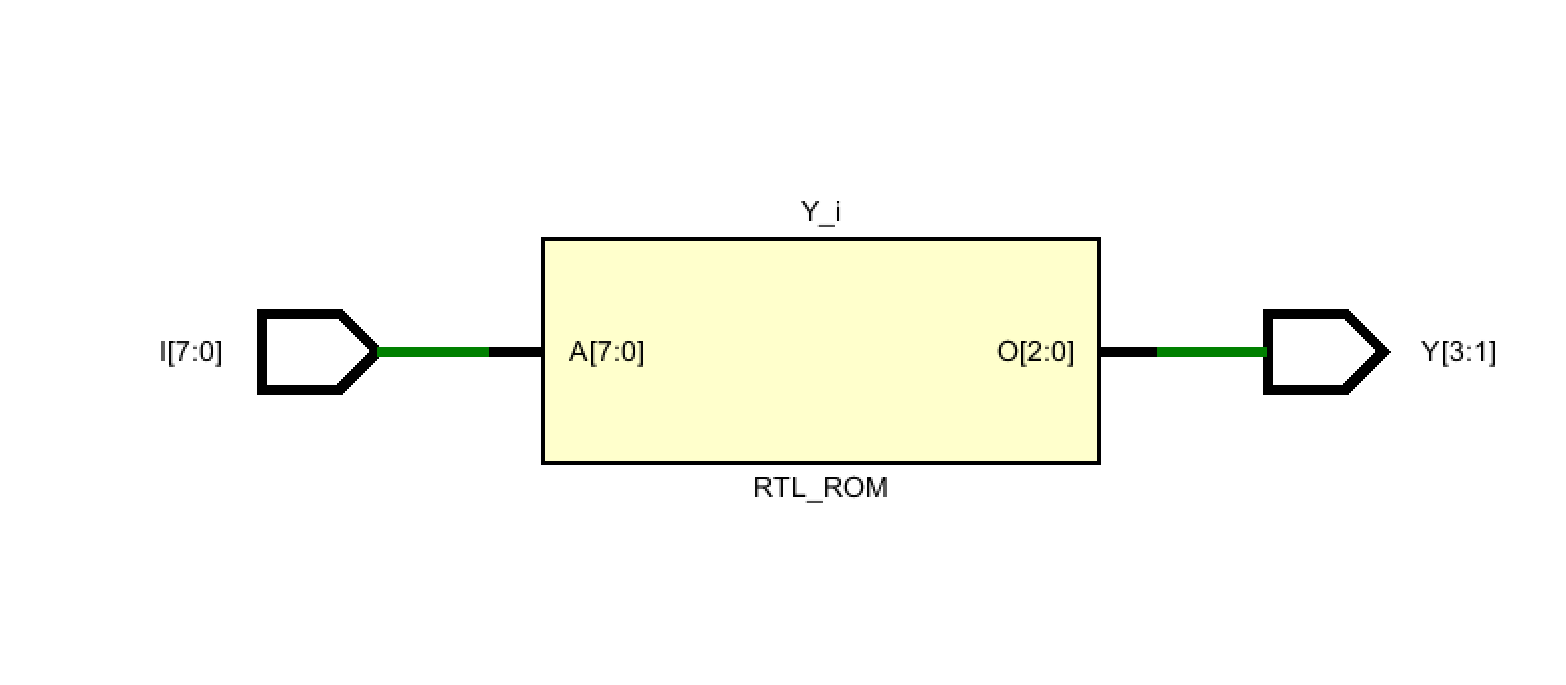
default : Y=3'b000;

endcase

end

endmodule

仿真图：



代码分析：

首先声明设计模块名称pre\_encoder83\_case，声明参数变量I,Y。

然后声明I为8位线输入，输出为3位寄存器。

使用always@(\*)，表明每当有输入变量发生变化时，都要更新输出值。 使用casex()语句来对输入信号I进行判断，并根据不同的I值对输出Y进行更新。补充说明casex(I)模块中，判断中的X表示无关项，即取0或取1都可以，无区别。

3.1.2 if结构的设计文件

module pre\_encoder83\_if(I,Y);

input I;

output Y;

wire[7:0] I;

reg[3:1]Y;

always @(\*) begin

if(I[7]==1)Y=3'b111;

else if(I[6]==1)Y=3'b110;

else if(I[5]==1)Y=3'b101;

else if(I[4]==1)Y=3'b100;

else if(I[3]==1)Y=3'b011;

else if(I[2]==1)Y=3'b010;

else if(I[1]==1)Y=3'b001;

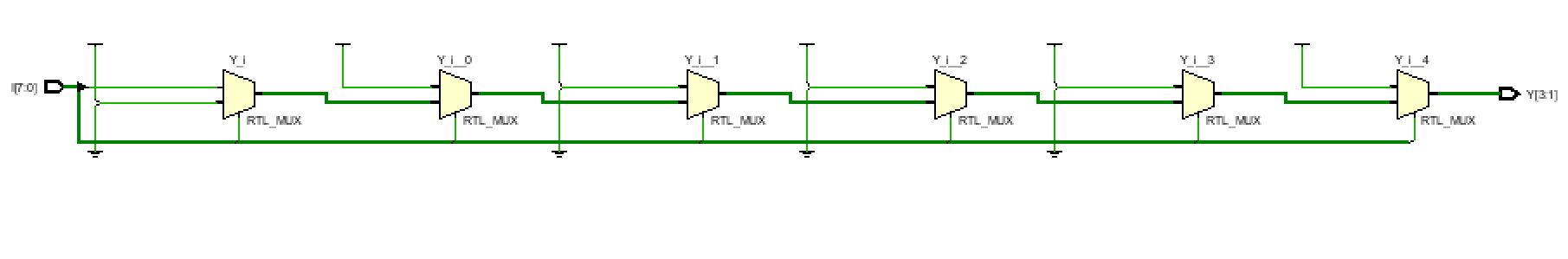
else if(I[0]==1)Y=3'b000;

else Y=3'b000;

end

endmodule

仿真图：



代码分析：

大部分与pre\_encoder83\_case相同，只有在更新值时有所不同：

不用case直接使用if,else if 语句。需要注意的是，由于是高位优先编码器，应该先判断高位，后判断低位，才能保证高位优先。即只有最高的为1的为有效位，比其低的位如果为1仍然为无效位，没有影响

3.2 83编码器的仿真文件

module sim\_encoder83();

reg[7:0] x;

wire[2:0] y\_assign,y\_case,y\_pre\_case,y\_pre\_if;

integer i;

initial begin

x=1;

for(i=0;i<7;i=i+1)#10 x=x\*2;

#10 x=128;

while(x>0)#5 x=x-1;

end

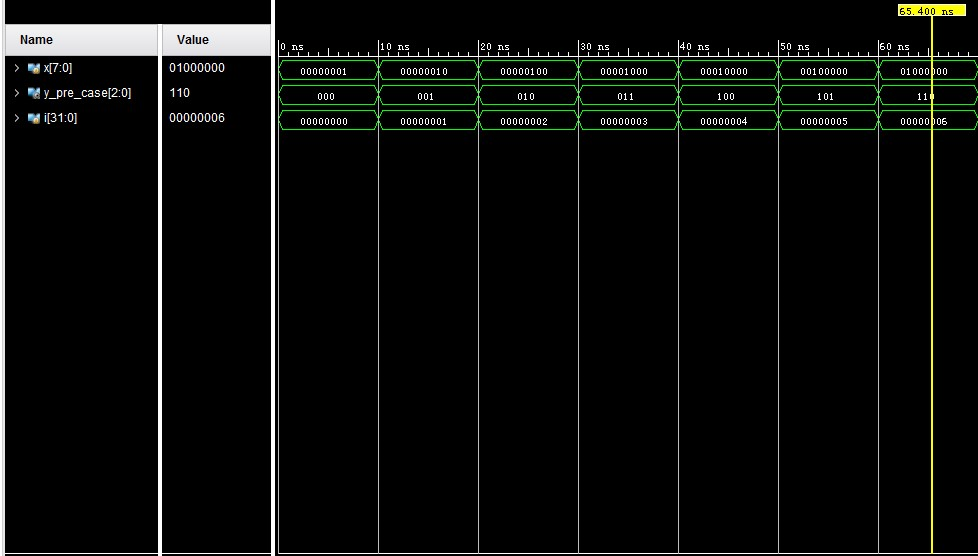
//encoder83\_assign encoder83\_assign\_1(x,y\_assign);

//encoder83\_case encoder83\_case\_1(x,y\_case);

pre\_encoder83\_case pre\_encoder83\_case\_1(.I(x),.Y(y\_pre\_case));

pre\_encoder83\_if pre\_encoder83\_if\_1(.I(x),.Y(y\_pre\_if));

endmodule

 时序图：

代码分析：

仿真文件按ppt给出，基本步骤步骤与之前相同，不再赘述。

不同之处在于讲8位变量x直接按数值统一操作，而不是分别操作每一位。X的每一位是x值的8位二进制对应的位。

3.3 代码分析

4、38译码器的设计和仿真

4.1 38译码器设计文件：

###decoder38.v###

module decoder38(I,Y);

input I;

output Y;

wire[2:0] I;

reg[7:0] Y;

always @(I) begin

case(I)

3'b000: Y=8'b0000\_0001;

3'b001: Y=8'b0000\_0010;

3'b010: Y=8'b0000\_0100;

3'b011: Y=8'b0000\_1000;

3'b100: Y=8'b0001\_0000;

3'b101: Y=8'b0010\_0000;

3'b110: Y=8'b0100\_0000;

3'b111: Y=8'b1000\_0000;

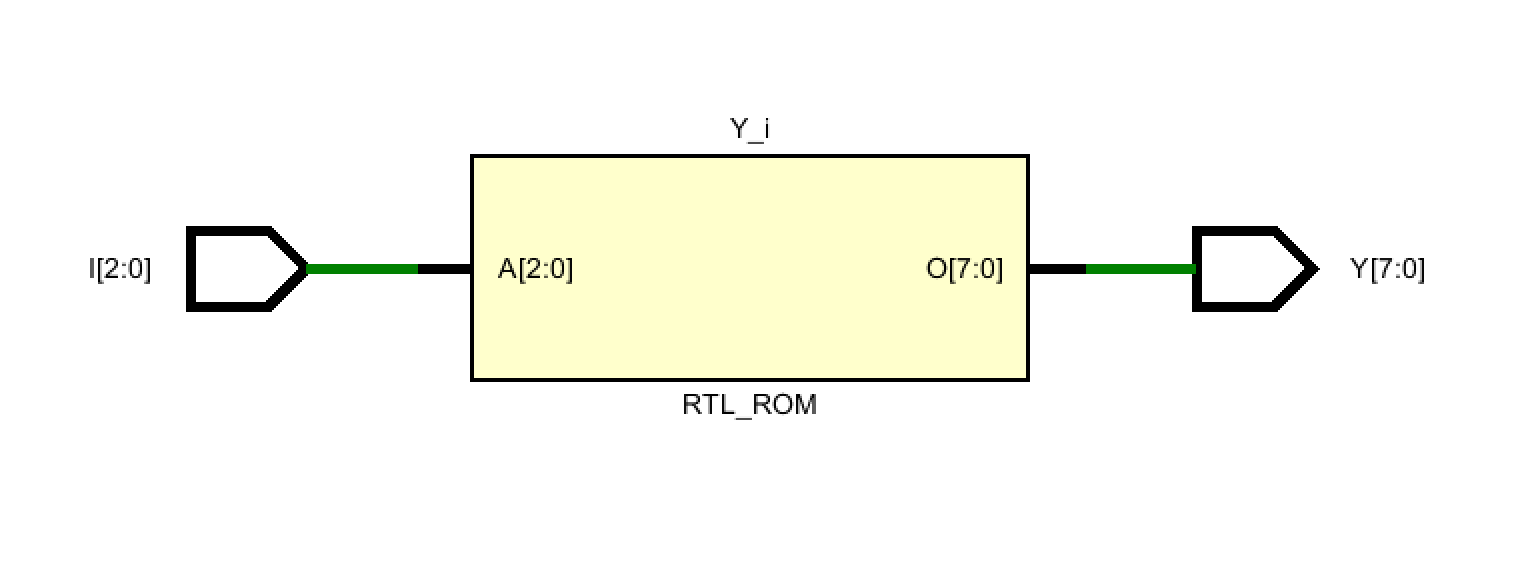
default: Y=8'b0000\_0001;

endcase

end

endmodule

仿真图：



代码分析：

基础操作除文件名、输入输出信号位数需调整外，与83编码器基本相同，不再赘述。

我选择使用case语句进行设计。3’b000表示3位输入信号I的二进制表示为000，此时译码信号输出为8’b0000\_0000，即8位二进制0，其他情况同理。

4.2 38编码器仿真文件

###sim\_decoder38.v###

module sim\_decoder38();

reg[2:0] x;

wire[7:0] y;

integer i;

initial begin

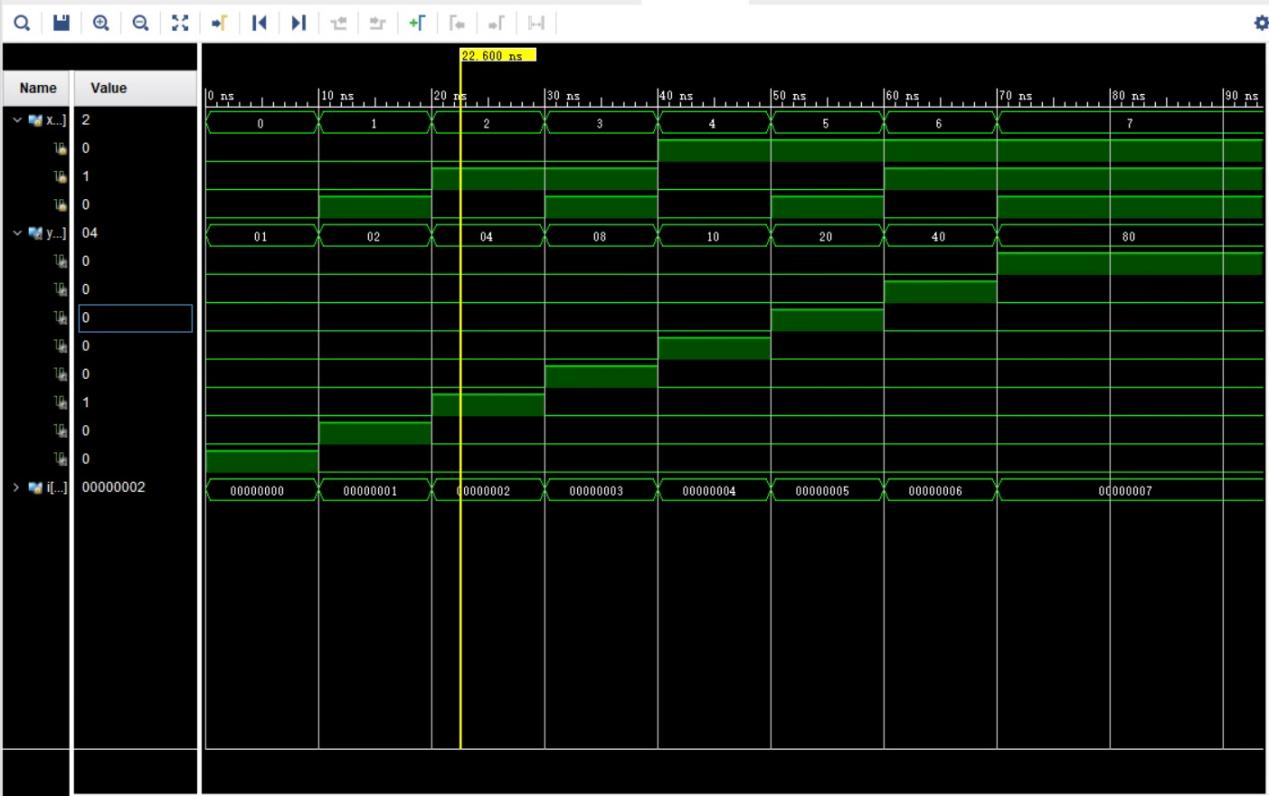
x=0;

for(i=0;i<7;i=i+1)#10 x=x+1;

end

decoder38 decoder(x,y);

endmodule

时序图：

代码分析：首先声明声明仿真文件模块名称sim\_decoder38，声明3位寄存器x，8位线性变量y，变量i，x初始化为0，使用for语句，每隔10ns对x实行加一操作，保证出现所有输入情况。最后实例化decoder模块。

**五、调试和心得体会**

本次实验在实例化模块时，对应名称赋值时为带.造成错误（例如把(.I(x),.Y(y\_pre\_if))错误写成(I(x),Y(y\_pre\_if))，多次调试后才发生错误。

心得体会：本次实验让我更加熟悉了Verilog的各种语句。更加深入的认识到了Verilog按模块编程的思想。认识到了reg变量和wire变量之间是可以相互转换的。除此之外，我还认识到可以直接给多位变量赋值，如给 reg[2:0] x=5,即把x更新为5的3位2进制表达。总之，这次实验让我收获很多。